

# Reducing and monitoring round-off error propagation for symplectic implicit Runge-Kutta schemes

Mikel Antoñana, Joseba Makazaga, Ander Murua  
KZAA saila, Informatika Fakultatea, UPV/EHU  
Donostia / San Sebastián

## Abstract

We propose an implementation of symplectic implicit Runge-Kutta schemes for highly accurate numerical integration of non-stiff Hamiltonian systems based on fixed point iteration. Provided that the computations are done in a given floating point arithmetic, the precision of the results is limited by round-off error propagation. We claim that our implementation with fixed point iteration is near-optimal with respect to round-off error propagation under the assumption that the function that evaluates the right-hand side of the differential equations is implemented with machine numbers (of the prescribed floating point arithmetic) as input and output. In addition, we present a simple procedure to estimate the round-off error propagation by means of a slightly less precise second numerical integration. Some numerical experiments are reported to illustrate the round-off error propagation properties of the proposed implementation.

## 1 Introduction

When numerically integrating an autonomous Hamiltonian system, one typically monitors the error in the preservation of the Hamiltonian function to check the precision of the numerical solution. However, severe loss of precision can actually occur for sufficiently long integration intervals, while displaying a good preservation of the value of Hamiltonian function. For high precision numerical integrations, where round-off errors may dominate truncation errors, it is highly desirable both reducing and monitoring the propagation of round-off errors.

We propose an implementation of symplectic implicit Runge-Kutta schemes (such as RK collocation methods with Gaussian nodes) that takes special care in reducing the propagation of round-off errors. Our implementation is intended to be applied to non-stiff problems, which motivates us to solve the implicit equations by fixed-point iteration (see for instance [10][4] for numerical tests comparing the efficiency of implementations based on fixed point iterations and simplified Newton).

We work under the assumption that the (user defined) function that evaluates the right-hand side of the differential equations is implemented in such a way that input and output arguments are machine numbers of some prescribed floating point arithmetic. Our actual implementation includes the option of computing, in addition to the numerical solution, an estimation of the propagated round-off error.

The starting point of our implementation is the work of Hairer et al. [5]. There, the authors observe that a standard fixed-point implementation of symplectic implicit RK (applied with compensated summation [6]) exhibits an unexpected systematic error in energy due to round-off errors, not observed in explicit symplectic methods. They make the following observations that allow them to understand that unfavorable error behavior: (a) The implicit Runge-Kutta method whose coefficients  $\tilde{b}_i, \tilde{a}_{ij}$  are the floating-point representation of the coefficient  $b_i, a_{i,j}$  of a symplectic Runge-Kutta method is not symplectic; (b) The error due to the application at each step of a fixed point iteration with standard stopping criterion (depending on a prescribed tolerance of the iteration error) tends to have a systematic character. Motivated by these observations, they modify the standard implementation of fixed point iteration which allows them to reduce the effect of round-off errors. No systematic error in energy is observed in the numerical experiments reported in [5]. However, we observe in some numerical experiments that the stopping criterion for the fixed point iteration that they propose fails to work properly in some cases. In addition, we claim that their implementation is still not optimal with respect to round-off error propagation.

In Section 3, we propose alternative modifications of the standard fixed point implementation of symplectic implicit Runge-Kutta methods, which compare very favorably with that proposed in [5].

We first define a reference implementation with fixed point iteration where all the arithmetic operations other than the evaluation of the right-hand side of the system of differential equations are performed in exact arithmetic, and as many iterations as needed are performed in each step. Such an implementation, that we call FPIEA (Fixed Point iteration with Exact Arithmetic) implementation, is based on the following two modifications to the standard implementation with fixed point iterations: (i) From one hand, we reformulate each symplectic implicit Runge-Kutta method in such a way that its coefficients can be approximated by machine numbers while still keeping its symplectic character exactly (Subsection 3.1). (ii) On the other hand, we propose a modification of the stopping criterion introduced in [5] that is more robust and is independent of the chosen norm (Subsection 3.2)

The implementation we present here is based on the FPIEA implementation, with most multiplications and additions performed (for efficiency reasons) in the prescribed floating point arithmetic, but some of the operations performed with special care in order to reduce the effect of round-off errors. In particular, this includes a somewhat non-standard application of Kahan's "compensated summation" algorithm [7][6][9], described in detail in Subsection 3.3.

Finally, in Subsection 3.4, we present a simple procedure to estimate the

round-off error propagation as the difference of the actual numerical solution, and a slightly less precise second numerical solution. These two numerical solutions can be computed either in parallel, or sequentially with a lower computational cost than two integrations executed in completely independent way.

In Section 4, we show some numerical experiments to assess the performance of our final implementation. Some concluding remarks are presented in Section 5.

## 2 Preliminaries

### 2.1 Numerical integration of ODEs by symplectic IRK schemes

We are mainly interested in the application of symplectic implicit Runge-Kutta (IRK) methods for the numerical integration of Hamiltonian systems of the form

$$\frac{d}{dt}q^j = \frac{\partial H(p, q)}{\partial p^j}, \quad \frac{d}{dt}p^j = -\frac{\partial H(p, q)}{\partial q^j}, \quad j = 1, \dots, d, \quad (1)$$

where  $H : \mathbb{R}^{2d} \rightarrow \mathbb{R}$ . Recall that the Hamiltonian function  $H(q, p)$  is a conserved quantity of the system.

More generally, we consider initial value problems of systems of autonomous ODEs of the form

$$\frac{d}{dt}y = f(y), \quad y(t_0) = y_0, \quad (2)$$

where  $f : \mathbb{R}^D \rightarrow \mathbb{R}^D$  is a sufficiently smooth map and  $y_0 \in \mathbb{R}^D$ . In the case of the Hamiltonian system (1),  $D = 2d$ ,  $y = (q^1, \dots, q^d, p^1, \dots, p^d)$ .

For the system of differential equations (2), an  $s$ -stage implicit Runge-Kutta method is determined by an integer  $s$  and the real coefficients  $a_{ij}$  ( $1 \leq i, j \leq s$ ),  $b_i$  ( $1 \leq i \leq s$ ). The approximations  $y_n \approx y(t_n)$  to the solution  $y(t)$  of (2) at  $t = t_n = t_0 + nh$  for  $n = 1, 2, 3, \dots$  are computed as

$$y_{n+1} = y_n + h \sum_{i=1}^s b_i f(Y_{n,i}), \quad (3)$$

where the *stage vectors*  $Y_{n,i}$  are implicitly defined at each step by

$$Y_{n,i} = y_n + h \sum_{j=1}^s a_{ij} f(Y_{n,j}), \quad i = 1, \dots, s. \quad (4)$$

An IRK scheme is symplectic if and only if [10]

$$b_i a_{ij} + b_j a_{ji} - b_i b_j = 0, \quad 1 \leq i, j \leq s. \quad (5)$$

In that case, the IRK scheme conserves exactly all quadratic first integrals of the original system (2), and if the system is Hamiltonian, under certain assumptions [4], it approximately conserves the value of the Hamiltonian function  $H(y)$  over long time intervals.

## 2.2 Floating point version of an IRK integrator

Let  $\mathbb{F} \subset \mathbb{R}$  be the set of machine numbers of a prescribed floating point system. Let  $\text{fl} : \mathbb{R} \rightarrow \mathbb{F}$  be a map that sends each real number  $x$  to a nearest machine number  $\text{fl}(x) \in \mathbb{F}$ .

We assume that instead of the original map  $f : \mathbb{R}^D \rightarrow \mathbb{R}^D$ , we have a computational substitute

$$\tilde{f} : \mathbb{F}^D \rightarrow \mathbb{F}^D. \quad (6)$$

Ideally, for each  $x \in \mathbb{F}^D$ ,  $\tilde{f}(x) := \text{fl}(f(x))$ . In practice, the intermediate computations to evaluate  $\tilde{f}$  are typically made using the floating point arithmetic corresponding to  $\mathbb{F}$ , which will result in some error  $\|\tilde{f}(x) - \text{fl}(f(x))\|$  caused by the accumulated effect of several round-off errors.

We aim at efficiently implementing a given symplectic IRK scheme under the assumption that  $f : \mathbb{R}^D \rightarrow \mathbb{R}^D$  is replaced by (6). Hence, the effect of round-off errors will be present even in the best possible ideal implementation where exact arithmetic were used for all the computations except for the evaluations of the map (6). Our goal is to implement the IRK scheme working at the prescribed floating point arithmetic, in such a way that the effect of round-off errors is similar in nature and relatively close in magnitude to that of such ideal implementation.

## 2.3 Kahan's compensated summation algorithm

Obtaining the numerical approximation  $y_n \approx y(t_n)$ , ( $n = 1, 2, \dots$ ) to the solution  $y(t)$  of the initial value problem (2) defined by (3)–(4) requires computing the sums

$$y_{n+1} = y_n + x_n, \quad n = 0, 1, 2, \dots, \quad (7)$$

where

$$x_n = h \sum_{i=1}^s b_i f(Y_{n,i}).$$

For an actual implementation that only uses a floating point arithmetic with machine numbers in  $\mathbb{F}$ , special care must be taken with the additions (7). It is worth mentioning that for sufficiently small step-length  $h$ , the components of  $x_n$  are smaller in size than those of  $y_n$  (provided that the components of the solution  $y(t)$  of (2) remain away from zero). The naive recursive algorithm  $\hat{y}_{n+1} := \text{fl}(\hat{y}_n + \text{fl}(x_n))$ , ( $n = 0, 1, 2, 3, \dots$ ), typically suffers, for large  $n$ , a significant loss of precision due to round-off errors. It is well known that such a round-off error accumulation can be greatly reduced with the use of Kahan's compensated summation algorithm [7] (see also [6], [9]). Given a sequence  $\{y_0, x_0, x_1, \dots, x_n, \dots\} \subset \mathbb{F}$  of machine numbers, Kahan's algorithm is aimed to compute the sums  $y_n = y_0 + \sum_{\ell=0}^{n-1} x_\ell$ , ( $n \geq 1$ ), using a prescribed floating point arithmetic, more precisely than with the naive recursive algorithm. In Kahan's algorithm, machine numbers  $\tilde{y}_n$  representing the sums  $y_n$  are computed along with additional machine numbers  $e_n$  in-

tended to capture the error  $y_n - \tilde{y}_n$ . The actual algorithm reads as follows:

```

 $\tilde{y}_0 = y_0; e_0 = 0;$ 
for  $l \leftarrow 0$  to  $n$  do
     $X_l = \text{fl}(x_l + e_l);$ 
     $\tilde{y}_{l+1} = \text{fl}(\tilde{y}_l + X_l);$ 
     $\hat{X}_l = \text{fl}(\tilde{y}_{l+1} - \tilde{y}_l);$ 
     $e_{l+1} = \text{fl}(X_l - \hat{X}_l);$ 
end

```

**Algorithm 1:** Kahan's compensated summation.

The sums  $\tilde{y}_l + e_l$  (which in general do not belong to  $\mathbb{F}$ ) are more precise approximations of the exact sums  $y_l$  than  $\tilde{y}_l \in \mathbb{F}$ . In this sense, if  $y_0 \notin \mathbb{F}$ , the algorithm (1) should be initialized as  $\tilde{y}_0 := \text{fl}(y_0)$  and  $e_0 := \text{fl}(y_0 - \tilde{y}_0)$  (rather than  $e_0 = 0$ ).

Of course, algorithm (1) also makes sense for  $D$ -vectors of machine numbers  $\tilde{y}_0, e_0, x_0, x_1, \dots, x_n \in \mathbb{F}^D$ . In this setting, algorithm (1) can be interpreted as a family of maps parametrized by  $n$  and  $D$ ,

$$S_{n,D} : \mathbb{F}^{(n+3)D} \rightarrow \mathbb{F}^{2D},$$

that given the arguments  $\tilde{y}_0, e_0, x_0, x_1, \dots, x_n \in \mathbb{F}^D$ , returns  $\tilde{y}_{n+1}, e_{n+1} \in \mathbb{F}^D$  such that  $\tilde{y}_{n+1} + e_{n+1}$  is intended to represent the sum  $(\tilde{y}_0 + e_0) + x_0 + x_1 + \dots + x_n$  (with some small error).

### 3 Proposed implementation of symplectic IRK schemes

#### 3.1 Symplectic schemes with machine number coefficients

If the coefficients  $b_i, a_{ij}$  determining a symplectic IRK are replaced by machine numbers  $\tilde{b}_i, \tilde{a}_{ij} \in \mathbb{F}$  that approximate them (say,  $\tilde{b}_i := \text{fl}(b_i)$ ,  $\tilde{a}_{ij} := \text{fl}(a_{ij})$ ), then the resulting IRK scheme typically fails to satisfy the symplecticity conditions (5). This results in a method that does not conserve quadratic first integrals and exhibits a linear drift in the value of the Hamiltonian function [5].

Motivated by that, we recast the definition of a step of the IRK method as follows:

$$Y_{n,i} = y_n + \sum_{j=1}^s \mu_{ij} L_{n,j}, \quad L_{n,i} = hb_i f(Y_{n,i}), \quad i = 1, \dots, s, \quad (8)$$

$$y_{n+1} = y_n + \sum_{i=1}^s L_{n,i}, \quad (9)$$

where

$$\mu_{ij} = a_{ij}/b_j, \quad 1 \leq i, j \leq s.$$

Condition (5) now becomes,

$$\mu_{ij} + \mu_{ji} - 1 = 0, \quad 1 \leq i, j \leq s.$$

The main advantage of the proposed formulation over the standard one is that the absence of multiplications in the alternative symplecticity condition makes possible (see Appendix A for the particular case of the 12th order Gauss collocation IRK method) to find machine number approximations  $\tilde{\mu}_{ij}$  of  $\mu_{ij} = a_{ij}/b_j$  satisfying exactly the symplecticity condition

$$\tilde{\mu}_{ij} + \tilde{\mu}_{ji} - 1 = 0, \quad 1 \leq i, j \leq s. \quad (10)$$

### 3.2 Iterative solution of the nonlinear Runge-Kutta equations

The fixed point iteration can be used to approximately compute the solution of the implicit equations (8) as follows: For  $k = 1, 2, \dots$  obtain the approximations  $Y_{n,i}^{[k]}, L_{n,i}^{[k]}$  of  $Y_{n,i}, L_{n,i}$  ( $i = 1, \dots, s$ ) as

$$L_{n,i}^{[k]} = hb_i f(Y_{n,i}^{[k-1]}), \quad Y_{n,i}^{[k]} = y_n + \sum_{j=1}^s \mu_{ij} L_{n,j}^{[k]} \quad i = 1, \dots, s. \quad (11)$$

The iteration may be initialized simply with  $Y_i^{[0]} = y_n$ , or by some other procedure that uses the stage values of the previous steps [4]. If the step-length  $h$  is sufficiently small, these iterations converge to a fixed point that is solution of the algebraic equations (8).

The situation is different for an actual computational version of these iterations, where  $f$  is replaced in (11) by its computational substitute (6). The  $k$ th iteration then reads as follows: For  $i = 1, \dots, s$ ,

$$f_{n,i}^{[k]} = \tilde{f}(\mathfrak{fl}(Y_{n,i}^{[k-1]})), \quad L_{n,i}^{[k]} = hb_i f_{n,i}^{[k]}, \quad Y_{n,i}^{[k]} = y_n + \sum_{j=1}^s \mu_{ij} L_{n,j}^{[k]}. \quad (12)$$

In this case, either a fixed point of (12) is reached in a finite number of iterations, or the iteration fails (mathematically speaking) to converge. In the former case, however (provided that  $h$  is small enough for the original iteration (11) to converge), after a finite number of iterations, a computationally acceptable approximation to the fixed point of (11) is typically achieved, and the successive iterates remain close to it. According to our experience and the numerical experiments reported in [5], a computational fixed point is reached for most steps in a numerical integration with sufficiently small step-length  $h$ .

In standard implementations of implicit Runge-Kutta methods, one considers

$$\Delta^{[k]} = (Y_{n,1}^{[k]} - Y_{n,1}^{[k-1]}, \dots, Y_{n,s}^{[k]} - Y_{n,s}^{[k-1]}) \in \mathbb{R}^{sd},$$

(for notational simplicity, we avoid reflecting the dependence of  $n$  on  $\Delta^{[k]}$ ), and stops the iteration provided that  $\|\Delta^{[k]}\| \leq \text{tol}$ , with a prescribed vector norm

$\|\cdot\|$  and iteration error tolerance  $\text{tol}$ . If the chosen value of  $\text{tol}$  is too small, then the iteration may never end when the computational sequence does not arrive to a fixed point in a finite number of steps. If  $\text{tol}$  is not small enough, the iteration will stop too early, which will result in an iteration error of larger magnitude than round-off errors. Furthermore, as observed in [5], such iteration errors tend to accumulate in a systematic way.

The remedy proposed in [5] is to stop the iteration either if  $\Delta^{[k]} = 0$  (that is, if a fixed point is reached) or if  $\|\Delta^{[k]}\| \geq \|\Delta^{[k-1]}\|$ . The underlying idea is that (provided that  $h$  is small enough for the original iteration (11) to converge), typically  $\|\Delta^{[k]}\| < \|\Delta^{[k-1]}\|$  whenever the iteration error is substantially larger than round-off errors, and thus  $\|\Delta^{[k]}\| \geq \|\Delta^{[k-1]}\|$  may indicate that round-off errors are already significant.

We have observed that Hairer’s strategy works well in general, but in some cases it stops the iteration too early. Indeed, it works fine for the initial value problem on a simplified model of the outer solar system (OSS) reported in [5] with a step-size  $h$  of 500/3 days, but it goes wrong with  $h = 1000/3$ . Actually, we have run [Hairer’s fortran code](#) and observed that the computed numerical solution exhibits an error in energy that is considerably larger than round-off errors. The evolution of relative error in energy is displayed on the left of Figure 1, which shows a linear growth pattern. We have checked that, for instance, at the first step,

$$\|\Delta^{[1]}\| > \|\Delta^{[2]}\| > \dots > \|\Delta^{[12]}\| = 3.91 \times 10^{-14} \leq \|\Delta^{[13]}\| = 4.35 \times 10^{-14}$$

which causes the iteration to stop at the 13th iteration, which happens to be too early, since subsequently,  $\|\Delta^{[13]}\| > \|\Delta^{[14]}\| > \|\Delta^{[15]}\| > \|\Delta^{[16]}\| = 0$ .

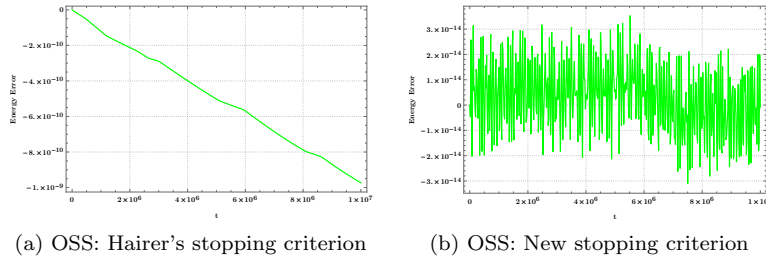


Figure 1: Evolution of relative error in energy for the outer solar system problem (OSS) with the original unperturbed initial values in [5] and doubled step-size ( $h = 1000/3$  days). (a) Hairer’s stopping criterion. (b) New stopping criterion.

Motivated by that we propose an alternative more stringent stopping criterion: Denote as  $\Delta_j^{[k]}$  the  $j$ th component ( $1 \leq j \leq sD$ ) of  $\Delta^{[k]} \in \mathbb{R}^{sD}$ . The fixed point iteration (12) is performed for  $k = 1, 2, \dots$  until either  $\Delta^{[k]} = 0$  or the following condition is fulfilled for two consecutive iterations:

$$\forall j \in \{1, \dots, sD\}, \quad \min \left( \{|\Delta_j^{[1]}|, \dots, |\Delta_j^{[k-1]}|\} \setminus \{0\} \right) \leq |\Delta_j^{[k]}|. \quad (13)$$

If  $K$  is the first positive integer such that (13) does hold for both  $k = K - 1$  and  $k = K$ , then we compute the approximation  $y_{n+1} \approx y(t_{n+1})$  as

$$y_{n+1} = y_n + \sum_{i=1}^s L_{n,i}^{[K]}.$$

The iteration typically stops with  $\Delta_j^{[K]} = 0$  for all  $j$ . However, when the iteration stops with  $\Delta_j^{[K]} \neq 0$  for some  $j$ , that is, when no computational fixed point is achieved, we still have to decide if this has been due to the effect of small round-off errors, or because the step-size is not small enough for the convergence of the iteration (11). Here is the only point where our implementation depends on a norm-based standard criterion (with rather loose absolute and relative error tolerances) to decide if  $\Delta^{[K]} \in \mathbb{F}^{sD}$  is small enough.

We have repeated the experiment of OSS with  $h = 1000/3$  by replacing Hairer's stopping criterion by our new one. The evolution of the resulting energy errors are displayed on the right of Figure 1.

### 3.3 Machine precision implementation of new algorithm

Subsections 3.1 and 3.2 completely determine the FPIEA (Fixed Point Iteration with Exact Arithmetic) implementation referred to in the Introduction. We next describe in detail our machine precision implementation of the algorithm described (for exact arithmetic) in previous subsection.

Consider appropriate approximations  $\tilde{b}_i \in \mathbb{F}$  of  $b_i$  ( $i = 1, \dots, s$ ), and let  $\tilde{\mu}_{ij} \in \mathbb{F}$  ( $i, j = 1, \dots, s$ ) be approximations of  $\mu_{ij}$  satisfying exactly the symplecticity condition (10).

Given  $y_n \in \mathbb{R}^D$ , we consider  $\tilde{y}_0 = \text{fl}(y_0)$  and  $e_0 = \text{fl}(y_n - \tilde{y}_n)$ . For each  $n = 0, 1, 2, \dots$ , we initialize  $Y_{n,i}^{[0]} = y_n$ , and successively compute for  $k = 1, 2, \dots$

$$\begin{aligned} f_{n,i}^{[k]} &= \tilde{f}(Y_{n,i}^{[k-1]}), & L_{n,i}^{[k]} &= \text{fl}(h \tilde{b}_i f_{n,i}^{[k]}), \\ Z_{n,i}^{[k]} &= e_n + \sum_{j=1}^s \tilde{\mu}_{ij} L_{n,j}^{[k]}, & Y_{n,i}^{[k]} &= \text{fl}(\tilde{y}_n + Z_{n,i}^{[k]}) \end{aligned} \quad (14)$$

until the iteration is stopped at  $k = K$  according to the criteria described in Subsection 3.2. Hence,  $K$  is the highest index  $k$  such that  $f_{n,i}^{[k]}$  has been computed.

In our actual implementation, one can optionally initialize  $Y_{n,i}^{[0]}$  by interpolating from the stage values of previous step, which improves the efficiency of the algorithm. Nevertheless, in all the numerical results reported Section 4 below, the simpler initialization  $Y_{n,i}^{[0]} = y_n$  is employed.

In (14), we evaluate each  $Z_{n,i}^{[k]} \in \mathbb{F}^D$  as

$$Z_{n,i}^{[k]} = (\dots((e_n + \tilde{\mu}_{i,1} L_{n,1}^{[k]}) + \tilde{\mu}_{i,2} L_{n,2}^{[k]}) + \dots + \tilde{\mu}_{i,s-1} L_{n,s-1}^{[k]}) + \tilde{\mu}_{i,s} L_{n,s}^{[k]}$$



where each multiplication and addition is performed in the prescribed floating point arithmetic.

We then compute  $\tilde{y}_{n+1}, e_{n+1} \in \mathbb{F}^D$  such that  $\tilde{y}_{n+1} + e_{n+1} \approx y(t_{n+1})$  as follows:

- compute for  $i = 1, \dots, s$  the vectors

$$E_{n,i} = h \tilde{b}_i f_{n,i}^{[K]} - L_{n,i}^{[K]}. \quad (15)$$

$$\delta_n = e_n + \sum_{i=1}^s E_{n,i}.$$

- finally, compute

$$(\tilde{y}_{n+1}, e_{n+1}) = S_{s,D}(\tilde{y}_n, \delta_n, L_{n,1}^{[K]}, \dots, L_{n,s}^{[K]}). \quad (16)$$

If the FMA (fused-multiply-add) instruction is available, it should be used to compute (15) (with precomputed coefficients  $h\tilde{b}_i$ ). The order in which the terms defining  $Z_n$  and  $\delta_n$  are actually computed in the floating point arithmetic is not relevant, as the corresponding round-off errors of the small corrections  $Z_n$  and  $\delta_n$  will have a very marginal effect in the computation of  $\tilde{y}_{n+1}$  and  $e_{n+1}$ .

### 3.4 Round-off error estimation.

We estimate the round-off error propagation of our numerical solution  $\tilde{y}_n + e_n \approx y(t_n)$  ( $n = 1, 2, \dots$ ) by computing its difference with a slightly less precise secondary numerical solution  $\hat{y}_n + \hat{e}_n \approx y(t_n)$  obtained with a modified version of the machine precision algorithm described in previous section. In this modified version of the algorithm, the components of each  $L_{n,i}^{[K]}$  in (16) are rounded to a machine number with a shorter mantissa. We next give some more details.

Let  $p$  be the number of binary digits of our floating point arithmetic. Given an integer  $r \geq 0$  and a machine number  $x$ , we define  $\text{fl}_{p-r}(x) := \text{fl}(2^r x + x) - 2^r x$ . This is essentially equivalent to rounding  $x$  to a floating point number with  $p-r$  significant binary digits.

We determine the algorithm for the secondary integration by fixing a positive integer  $r < p$  and modifying (16) in the implementation of the algorithm described in previous subsection as follows:

$$(\hat{y}_{n+1}, \hat{e}_{n+1}) = S_{s,D}(\hat{y}_n, \delta_n, \text{fl}_{p-r}(L_{n,1}^{[K]}), \dots, \text{fl}_{p-r}(L_{n,s}^{[K]})).$$

The proposed round-off error estimation can thus be obtained as the difference of the primary numerical solution and the secondary numerical solution obtained with a relatively small value of  $r$  (say,  $r = 3$ ). These two numerical solutions can be computed in parallel in a completely independent way.

In addition, we have implemented a sequential version with lower CPU requirements than two integrations executed sequentially in completely independent way. The key to do that is the following: At each step, the stage values  $Y_{n,i}$

( $i = 1, \dots, s$ ) of both primary and secondary integration will typically be very close to each other (as far as the estimated round-off error does not grow too much). Thus, the number of iterations of each step of the secondary integration can be reduced by using the final stage values  $Y_{n,i}$  ( $i = 1, \dots, s$ ) of the primary integration as initial values  $Y_{n,i}^{[0]}$  of the secondary integration.

## 4 Numerical experiments

We next report some numerical experiments to assess our implementation of the 6-stage Gauss collocation method of order 12 in the 64-bit IEEE double precision floating point arithmetic.

### 4.1 Test problems

We consider two different Hamiltonian problems corresponding to a double pendulum and the simulation of the outer solar system (considered in [4] and [5]) respectively. In all the cases, we consider a time-step  $h$  that is small enough for round-off errors to dominate over truncation errors.

#### 4.1.1 The double pendulum (DP) problem

We consider the planar double pendulum problem: a double bob pendulum with masses  $m_1$  and  $m_2$  attached by rigid massless rods of lengths  $l_1$  and  $l_2$ . This is a non-separable Hamiltonian system with two degrees of freedom, for which no explicit symplectic Runge-Kutta-type method is available, and hence Gauss collocation methods are a natural choice [8].

The configuration of the pendulum is described by two angles  $q = (\phi, \theta)$  (see figure 2): while  $\phi$  is the angle of the first bob, the second bob's angle is defined by  $\psi = \phi + \theta$ . We denote the corresponding momenta as  $p = (p_\phi, p_\theta)$ .

Its Hamiltonian function  $H(q, p)$  is

$$-\frac{l_1^2 (m_1 + m_2) p_\theta^2 + l_2^2 m_2 (p_\theta - p_\phi)^2 + 2 l_1 l_2 m_2 p_\theta (p_\theta - p_\phi) \cos(\theta)}{l_1^2 l_2^2 m_2 (-2 m_1 - m_2 + m_2 \cos(2\theta))} - g \cos(\phi) (l_1 (m_1 + m_2) + l_2 m_2 \cos(\theta)) + g l_2 m_2 \sin(\theta) \sin(\phi), \quad (17)$$

and we consider following parameters values

$$g = 9.8 \frac{m}{sec^2}, \quad l_1 = 1.0 \, m, \quad l_2 = 1.0 \, m, \quad m_1 = 1.0 \, kg, \quad m_2 = 1.0 \, kg.$$

We take two initial values from [2], the first one of non-chaotic character, and the second one exhibiting chaotic behaviour:

1. Non-Chaotic case (NCDP):  $q(0) = (1.1, -1.1)$  and  $p(0) = (2.7746, 2.7746)$ . We have integrated over  $T_{end} = 2^{12}$  seconds with step-size  $h = 2^{-7}$ . The numerical results will be sampled once every  $m = 2^{10}$  steps.

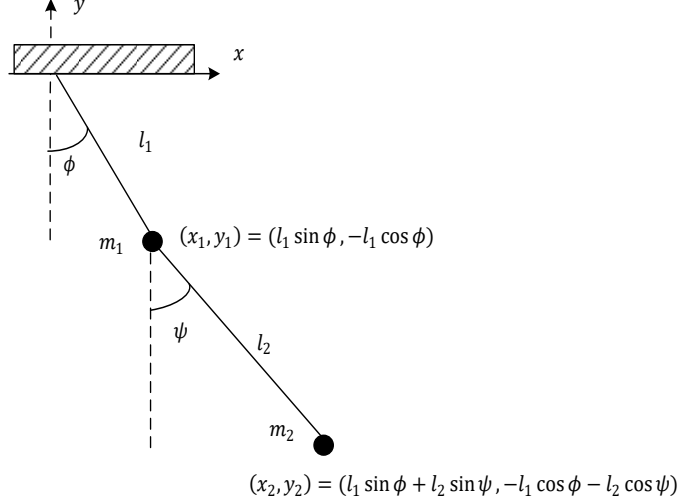


Figure 2: Double Pendulum.

2. Chaotic case (CDP):  $q(0) = (0, 0)$  and  $p(0) = (3.873, 3.873)$ . We have integrated over  $T_{end} = 2^8$  seconds with step-size  $h = 2^{-7}$ . We sample the numerical results once every  $m = 2^8$  steps.

Both initial value problems (NCDP and CDP) will be used to test the evolution of the global errors as well as to check the performance of the round-off error estimators. For the long term evolution of the errors in energy, only the NCDP problem will be considered.

#### 4.1.2 Simulation of the outer solar system (OSS)

We consider a simplified model of the outer solar system (sun, the four outer planets, and Pluto) under mutual gravitational (non-relativistic) interactions. This is a Hamiltonian system with 18-degrees of freedom ( $q_i, p_i \in \mathbb{R}^3$ ,  $i = 0, \dots, 5$ ) and Hamiltonian function is

$$H(q, p) = \frac{1}{2} \sum_{i=0}^N \frac{\|p_i\|^2}{m_i} - G \sum_{0 \leq i < j \leq N} \frac{m_i m_j}{\|q_i - q_j\|}. \quad (18)$$

We have considered the initial values and the values of the constant parameters ( $Gm_i$ ,  $i = 0, \dots, 5$ ) taken from [4, page 14]. We have integrated over  $T_{end} = 10^7$  days, with step-size  $h = 500/3$  and the numerical results are sampled once every  $m = 120$  steps.

Observe that (18) is a separable Hamiltonian, i.e., of the form  $H(p, q) = T(p) + U(q)$ . It is well known that the efficiency of standard fixed point iteration can be improved for Hamiltonians systems with separable Hamiltonian by considering a partitioned version of the fixed point iteration [4]. Nevertheless, as in [5], here we report the numerical results obtained with the standard non-partitioned fixed-point iteration. (We have actually checked that similar results are obtained with the partitioned version of the fixed point iteration, the main difference being that less iterations are performed at each step.)

## 4.2 Comparison of different sources of error in energy

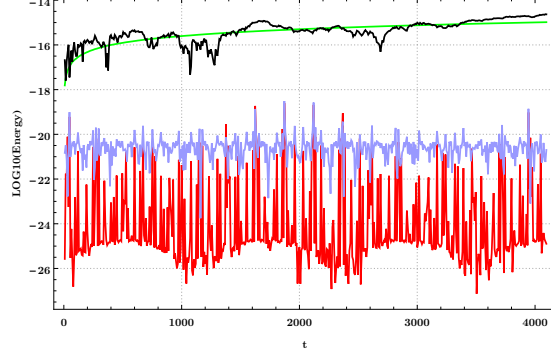
The error of a numerical solution  $\tilde{y}_n + e_n \approx y(t_n)$  ( $n = 1, 2, \dots$ ) computed with our double precision (DP) implementation of symplectic IRK schemes is a combined result of different kinds of errors:

1. The truncation error: The error due to replacing  $y(t_n)$ ,  $n = 1, 2, 3, \dots$  (where  $y(t)$  is the solution of the initial value problem (2)) by the numerical approximations  $y_n$  defined by (9)–(8) (with exact coefficients  $b_i, \mu_{ij}$ ).
2. The iteration error: In practice a finite number  $K$  of fixed point iterations (11) are applied, and the solution  $L_{n,i}, Y_{n,i}$  ( $i = 1, \dots, s$ ) of (8) are replaced by approximations  $L_{n,i}^{[K]}, Y_{n,i}^{[K]}$ . Then, in the FPIEA implementation, the corresponding numerical solution  $\bar{y}_{n+1}$  is computed at each step as

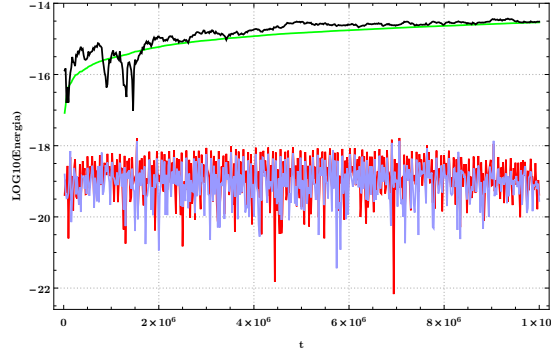
$$\bar{y}_{n+1} = y_n + \sum_{i=1}^s L_{n,i}^{[K]}.$$

3. The error due to replacing the original map  $f : \mathbb{R}^D \rightarrow \mathbb{R}^D$  by its computational substitute  $\tilde{f} : \mathbb{F}^D \rightarrow \mathbb{F}^D$ . This has a double effect: From one hand, in most steps, a computational fixed point is achieved in a finite number  $K$  of iterations, which causes an unavoidable iteration error. On the other hand, replacing  $f$  by  $\tilde{f}$  adds the effect of some round-off errors.
4. The error due to the application of a different IRK scheme. In our case, we apply the scheme (9)–(8) with  $b_i$  replaced by  $\tilde{b}_i \in \mathbb{F}$  ( $i = 1, \dots, s$ ) and each  $\mu_{ij}$  replace by double precision approximation  $\tilde{\mu}_{ij} \in \mathbb{F}$  satisfying condition (10).
5. The error due to using inexact arithmetic for the operations (other than the evaluation of  $\tilde{f}$ ) in the machine precision implementation of the algorithm.

We have simulated, for the double pendulum (the non-chaotic case, NCDP) and the outer solar system (OSS) respectively, the effect that each of the first four of such sources of errors has in the values of the energy (which of course is conserved in the exact solution) as follows:



(a) NCDP ( $h = 2^{-7}$ )



(b) OSS ( $h = 500/3$ )

Figure 3: We plot the evolution of energy error in logarithmic scale of the next algorithms implementations: A-algorithm as estimation of truncation error (red), B-algorithm as estimation of iteration error (green), C-algorithm as estimation of the effect of replacing the exact  $f$  by its double precision version  $\tilde{f}$  function (black) and D-algorithm as estimation of the effect of using double precision coefficients (blue).

- A. In order to estimate the truncation error, we have applied our algorithm fully implemented in quadruple precision.
- B. For the iteration error, we have applied the quadruple precision version of the algorithm modified so that the fixed point process in the  $n$ th step is stopped at the  $K$ th iteration provided that  $Y_{n,i}^{[K]}$  and  $Y_{n,i}^{[K-1]}$  coincide when rounded to double precision.
- C. In addition, we have estimated the effect (in the evolution of the energy) of the error due to replacing  $f$  by  $\tilde{f}$ , by considering the quadruple precision implementation of our algorithm with the double precision version of  $\tilde{f}$ .
- D. Finally, we have simulated the error due to the application of a RK scheme

with double precision coefficients by applying our quadruple precision implementation with double precision coefficients  $\tilde{b}_i, \tilde{\mu}_{ij}$ .

We next plot (Fig. 3), for each of the considered initial value problems, the evolutions of the energy errors corresponding to the items A–D in previous list. In both cases, we have chosen a step-size  $h$  such that truncation errors are smaller than round-off errors. We observe that the effect of using double precision coefficients ( $\tilde{b}_i, \tilde{\mu}_{ij}$ ) is also negligible compared to the propagation of round-off errors. The iteration error is similar in size to round-off errors.

### 4.3 Statistical analysis of errors

In order to make a more robust comparison of the numerical errors due to round-off errors, we adopt (as in [5]) an statistical approach. We have considered for each of the three initial value problem,  $P = 1000$  perturbed initial values by randomly perturbing each component of the initial values with a relative error of size  $\mathcal{O}(10^{-6})$ .

We will compare three different fixed point implementations of the 6-stage Gauss collocation method. In all of them, the same computational substitute  $\tilde{f} : \mathbb{F}^D \rightarrow \mathbb{F}^D$  is used instead of the original map  $f : \mathbb{R}^D \rightarrow \mathbb{R}^D$  defining the ODE (2):

1. The FPIEA (fixed point iteration with exact arithmetic) implementation, where the techniques described in Subsections 3.1 and 3.2 are applied to implement a fixed point iteration with all arithmetic operations (other than those used when evaluating  $\tilde{f}$ ) performed in exact arithmetic.
2. Our double precision version (coded in C) of the algorithm implemented in FPIEA. We will refer to it as DP (double precision).
3. The algorithm proposed in [5], implemented in [Hairer’s Fortran code](#).

From one hand, we want to check if the propagation of round-off errors in our DP implementation are qualitatively similar and close in magnitude to its exact arithmetic counterpart FPIEA. On the other hand, we want to see how our DP implementation compares with Hairer’s code.

In (Table 1) we display the percentage of steps that reach a computational fixed-point and the average number of fixed-point iterations per step in each of the referred three implementations when applied to two different initial value problems.

#### 4.3.1 Distribution of energy jumps

The local error in energy  $H(y_n) - H(y_{n-1})$  due to round-off errors, is "expected" to behave, for a good implementation free from biased errors, like an independent random variable. Then, provided that the numerical results are sampled every  $m$  steps, with a large enough sampling frequency  $m$ , an energy

Table 1: Percentage of steps that reach a computational fixed-point and the number of fixed-point iterations per step for the computations of non-chaotic double pendulum (NCDP), chaotic double pendulum (CDP), and the outer solar system (OSS) problems. In columns, we compare three different implementations: FPIEA, DP (double precision) and Hairer’s Fortran code.

	FPIEA		DP		Hairer	
NCDP	98.7%	8.5	98.8%	8.6	98.5%	8.6
CDP	98.9%	8.5	98.9%	8.6	98.4%	8.6
OSS	97.7%	14.4	97.4%	14.2	87.5%	14.1

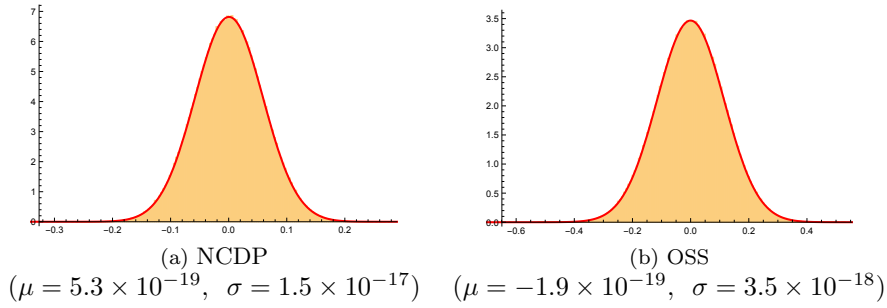


Figure 4: Histograms of  $KP$  samples of energy jumps of the DP implementation against the normal distribution  $N(\mu, \delta)$  for two problems (NCDP and OSS). The horizontal axis is multiplied by  $10^{15}$  and vertical axis indicates the frequency

jump  $H(y_{km}) - H(y_{mk-m})$  will behave as an independent variable with an approximately Gaussian distribution with mean  $\mu$  (ideally  $\mu = 0$ ) and standard deviation  $\sigma$ . So that the accumulated difference in energy,

$$H(y_{km}) - H(y_0) \quad (19)$$

at the sampled times  $t_{mk} = kmh$  would behave like a Gaussian random walk with standard deviation  $k^{\frac{1}{2}}\sigma = (t_{mk}/(mh))^{1/2}\sigma$ . This is sometimes referred to as Brouwer’s law in the scientific literature [3], from the original work on the accumulation of round-off errors in the numerical integration of Kepler’s problem done by Brouwer in [1].

In this sense, we want to check in which extent the (scaled) energy jumps,

$$(H(y_{km}) - H(y_{mk-m}))/H(y_0) \quad (20)$$

due to round-off errors after  $m$  steps approximately obey a Gaussian distribution in our double precision (DP) implementation.

If  $[0, T_{end}]$  is the integration interval, and  $P$  perturbed initial values are considered, we have a total number of  $KP$  samples of energy jumps, where  $K = T_{end}/(mh)$ . In Figure 4, we plot the histograms of  $KP$  samples of energy jumps obtained with our DP implementation against the normal distribution

$N(\mu, \delta)$  (where  $\mu$  and  $\sigma$  are the average and standard deviation of the samples respectively). For both initial value problems, non-chaotic double pendulum (NCDP), and the outer solar system (OSS), such histograms fits perfectly well to their corresponding normal distributions  $N(\mu, \delta)$ .

#### 4.3.2 Evolution of mean and standard deviation of errors

We next plot (Fig. 5) the evolution of the mean and standard deviation of the errors in energy of the FPIEA, DP, and Hairer's implementations for the NCDP and OSS problems respectively.

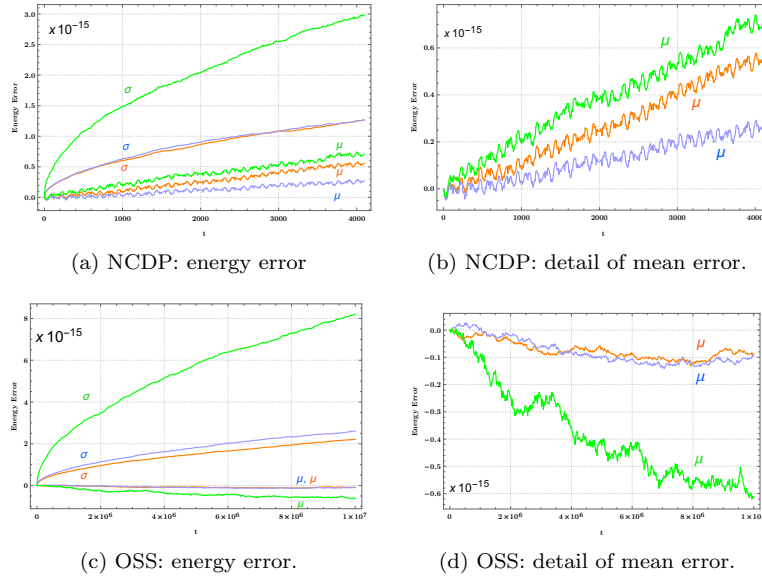


Figure 5: Evolution of mean ( $\mu$ ) and standard deviation ( $\sigma$ ) of errors in energy (left) and detail of the evolution of mean errors in energy (right), for DP implementation (blue), FPIEA implementation (orange), and Hairer's implementation (green). Non-Chaotic case (a,b), and outer solar system case (c, d)

Recall that FPIEA represents the best possible fixed point implementation of the IRK scheme provided that the double precision version  $\tilde{f}$  of the original  $f$  is used. We stress that we have made the stopping criterion of the FPIEA implementation even more stringent than in the DP implementation: we stop the fixed point iteration if either  $\Delta^{[k]} = 0$  or (13) is fulfilled during ten consecutive iterations. This way, we try to avoid the persistence of iteration errors in the case of steps where no computational fixed point is obtained. (Observe that whenever  $\Delta^{[k]} = 0$ , there is no point in performing more fixed point iterations, as in that case a computational fixed point has been achieved.)

The numerical tests in Figure 5 seen to confirm that our DP implementation is near optimal (that is, close to the FPIEA implementation), both with respect



to the standard deviation and the mean of the errors in energy.

We believe that some small linear drift of the mean energy error may be unavoidable for the fixed point implementations of IRK schemes in some cases (such as the NCDP example). This is consistent with the observation that the simulated iteration error displaying in Figure 1 is close in magnitude to the effect of round-off errors.

This is not of course inherent to the symplectic IRK scheme itself. In Figure 6, we display the results obtained for the NCDP example with a preliminary implementation of a simplified Newton implementation of the same IRK scheme as above. No linear drift of the mean energy error seems to appear.

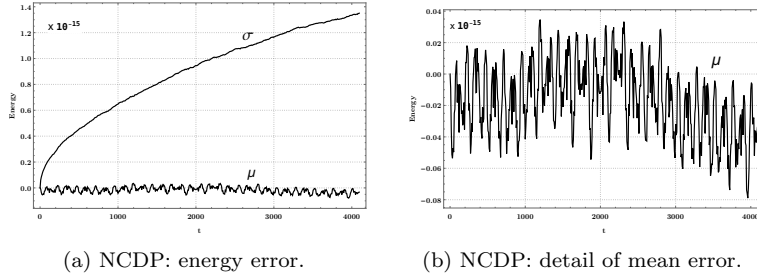
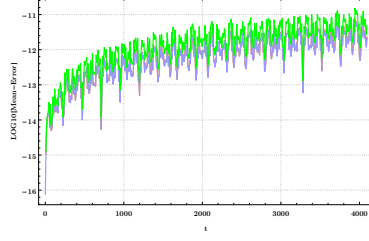


Figure 6: Evolution of mean ( $\mu$ ) and standard deviation ( $\sigma$ ) of errors in energy of a IRK implementation with simplified Newton iterations

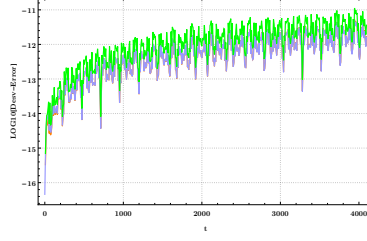
To end with the present subsection, we plot (Fig. 7) the evolution of the (mean and standard deviation of) errors in position of the FPIEA, DP, and Hairer's implementations for the NCDP, CDP and OSS problems respectively. The displayed results seem to confirm our claim of the DP implementation being a close-to-optimal fixed point implementation of the symplectic IRK scheme.

#### 4.4 Round-off error estimation

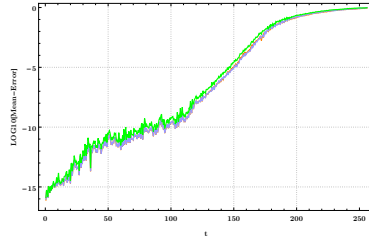
In order to assess the quality of the error estimation technique proposed in Subsection 3.4, we represent, In Fig. 8, for each of the three considered initial value problems (with the original unperturbed initial values), the evolution of the global errors in position of our DP implementation, together with the evolution of the estimations produced by using our technique applied with  $r = 3$ . In addition, we present for each of the three considered examples, the evolution of the mean error in positions of the application of our DP algorithm to  $P = 1000$  perturbed initial value problems, together with the evolution of the mean of the estimated errors in positions. We believe that the results indicate that the proposed round-off error estimation procedure is useful for the purpose of assessing the propagation of round-off errors.



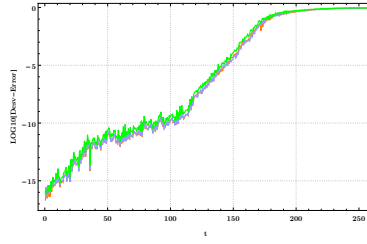
(a) NCDP: mean global error.



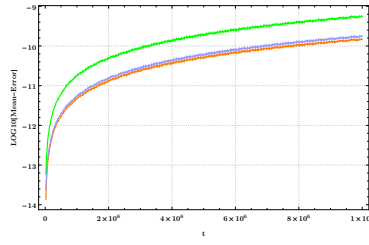
(b) NCDP: standard deviation global error.



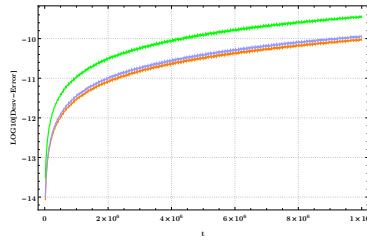
(c) CDP: mean global error.



(d) CDP: standard deviation global error.



(e) OSS: mean global error.



(f) OSS: standard deviation global error.

Figure 7: Evolutions of Mean (left) and standard deviation (right) of global errors in positions of DP implementation (blue), FPIEA implementation (orange), and Hairer's implementation (green): NCDP (a,b), CDP (c,d) and OSS (e, f)

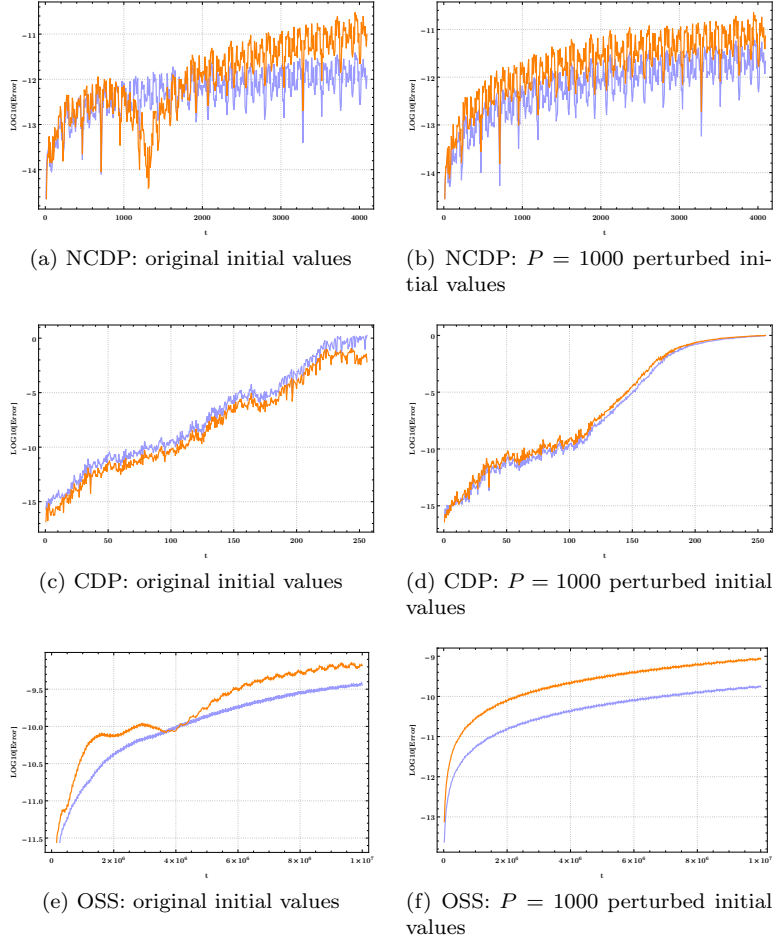


Figure 8: Left: estimation of the round-off error with the original unperturbed initial values. We compare the evolution of our error estimation (orange) with the evolution of the global error (blue). Right: evolution of the mean error in positions (blue) of the application of our DP algorithm to  $P = 1000$  perturbed initial value problems, together with the evolution of the mean of the estimated errors in positions (orange).

## 5 Concluding remarks

Symplectic implicit Runge-Kutta schemes (such as RK collocation methods with Gaussian nodes) are very appropriate for the accurate numerical integration of general Hamiltonian systems. For non-stiff problems, implementations based on fixed-point iterations seem to be more efficient than those based on Newton method or some of its variants.

We propose an implementation that takes special care in reducing the propagation of round-off errors, and includes the option of computing, in addition to the numerical solution, an estimation of the propagated round-off error. We claim that our implementation with fixed point iterations is near optimal, in the sense that the propagation of round-off errors is essentially no worse than the best possible implementation with fixed point iteration. Our claim seems to be confirmed by our numerical experiments.

A key point in our implementation has been the introduction of a new stopping criterion for the fixed point iteration. We believe that such a stopping criterion could be also useful in other contexts.

According to our numerical experiments, it seems that, in some cases, some small linear drift of the mean energy error may be unavoidable for the fixed point implementations of IRK schemes. Whenever avoiding any drift of energy error becomes critical it might be preferable to use some Newton based iteration instead.

The C code of our implicit Runge-Kutta implementation with fixed point iterations can be downloaded from [IRK-FixedPoint](https://github.com/mikelehu/IRK-FixedPoint) Github software repository or go to the next url: <https://github.com/mikelehu/IRK-FixedPoint>.

**Acknowledgements** M. Antoñana, J. Makazaga, and A. Murua have been partially supported by projects MTM2013-46553-C3-2-P from Ministerio de Economía y Comercio, Spain, by project MTM2016-76329-R “IMAGEARTH” from Spanish Ministry of Economy and Competitiveness and as part of the Consolidated Research Group IT649-13 by the Basque Government.

## References

- [1] D. Brouwer. [On the accumulation of errors in numerical integration](#). *The Astronomical Journal*, 46:149–153, 1937.
- [2] N. D. Dumitru. [Fast detection of chaotic or regular behavior of double pendulum system: application of the fast norm vector indicator method](#). *SEECCM III (South East European Conference on Computational Mechanics)*, 2013.
- [3] K.R. Grazier, W.I. Newman, J. M. Hyman, P. W. Sharp, and D. J. Goldstein. [Achieving brouwer’s law with high-order stormer multistep methods](#). *ANZIAM Journal*, 46:786–804, 2005.

- [4] E. Hairer, C. Lubich, and G. Wanner. *Geometric numerical integration: structure-preserving algorithms for ordinary differential equations*, volume 31. Springer Science & Business Media, 2006.
- [5] E. Hairer, R. I. McLachlan, and A. Razakarivony. *Achieving brouwer’s law with implicit runge–kutta methods*. *BIT Numerical Mathematics*, 48(2): 231–243, 2008.
- [6] Nicholas J. Higham. *Accuracy and stability of numerical algorithms*. Siam, 2002.
- [7] W. Kahan. Further remarks on reducing truncation errors. *Communications of the ACM*, 8(1):40, 1965.
- [8] R. I. McLachlan and P. Atela. *The accuracy of symplectic integrators*. *Nonlinearity*, 5(2):541, 1992.
- [9] J.M. Muller, N. Brisebarre, F. De Dinechin, C.P. Jeannerod, V. Lefevre, G. Melquiond, N. Revol, D. Stehlé, and S. Torres. *Handbook of floating-point arithmetic*. Springer Science & Business Media, 2009.
- [10] J.M. Sanz Serna and M.P. Calvo. *Numerical Hamiltonian problems*. Chapman and Hall, 1994.

## Appendix A: Computation of coefficients for 12th order Gauss collocation method

We next illustrate, by considering in detail the case of the 6th stage Gauss collocation method for the 64-bit IEEE double precision floating point arithmetic, how to determine appropriate machine number coefficients  $\mu_{ij}$ ,  $1 \leq i, j \leq s$ , that approximate the real numbers  $a_{ij}/b_j$  of a given symplectic integration.

For all  $i = 1, \dots, s$ ,  $\mu_{i,i} = 1/2$ . For  $1 \leq j < i \leq s$ ,  $\mu_{ij} := \text{fl}(a_{ij}/b_j)$ , which satisfy  $1/2 < |\mu_{ij}| < 2$ , which implies that  $\mu_{ji} := 1 - \mu_{ij}$  is a machine number. This results in machine number coefficients  $\mu_{ij}$  that satisfy the symplecticity conditions (10).

Given  $h$ , the coefficients  $hb_i = h \times b_i$  are precomputed as follows: for  $i = 2, \dots, s-1$ ,  $hb_i := \text{fl}(h \times b_i)$ , and

$$hb_1 := hb_s := (h - \sum_{i=2}^s hb_i)/2.$$